Team T

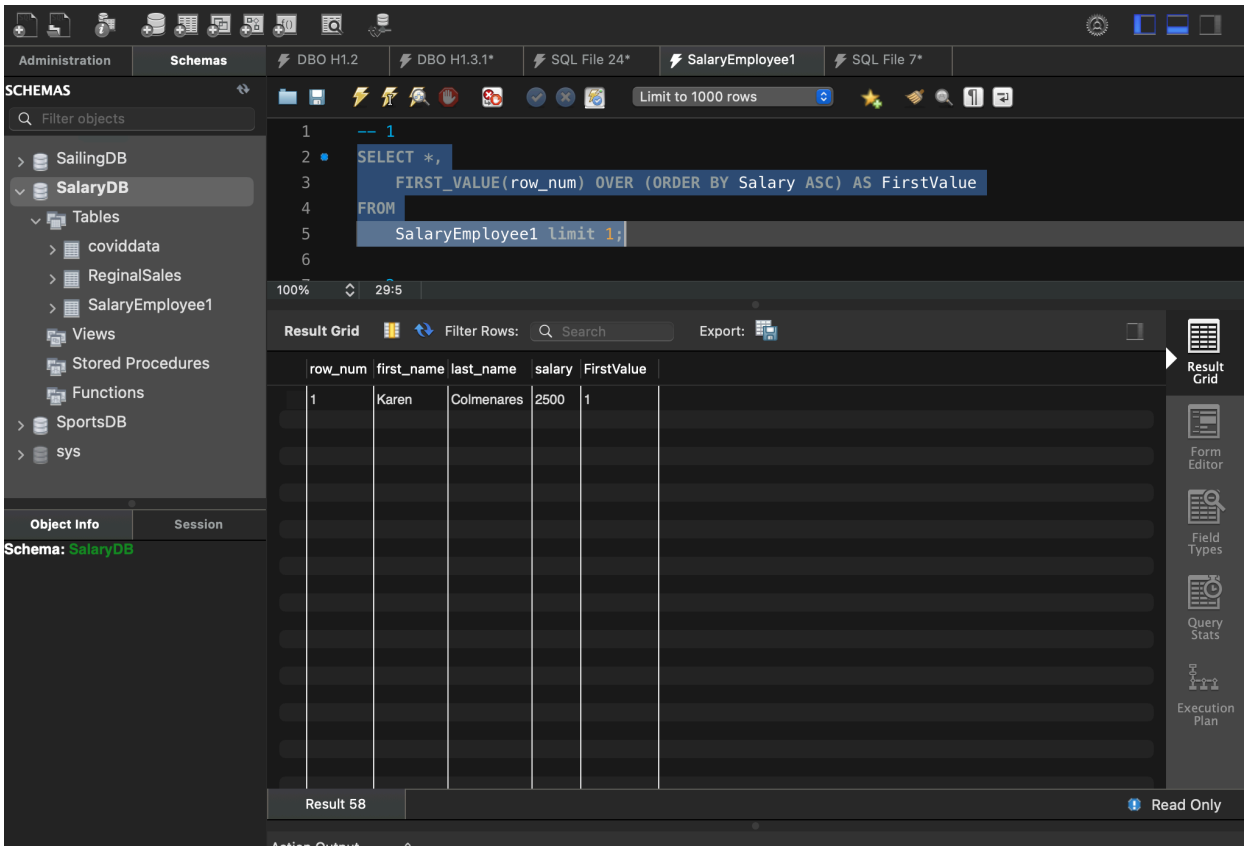
1. Hemanth Thathireddy (A20525346)
2. Satya Dineswara Reddy Satti (A20528639)
3. Sai Manasa Basani (A20544372)
4. Alavalapati Veera manohar Reddy (A20526070)
5. Ying Zhang (A20547843)

Part 1 : Advanced window functions

1. **SQL Statement:** Write a query to compute for the **FIRST_VALUE()** given the above dataset and return the value along with the entire row.

   SQL Query: SELECT *,
   FIRST_VALUE(row_num) OVER (ORDER BY Salary ASC) AS FirstValue
   FROM
   SalaryEmployee1 limit 1;

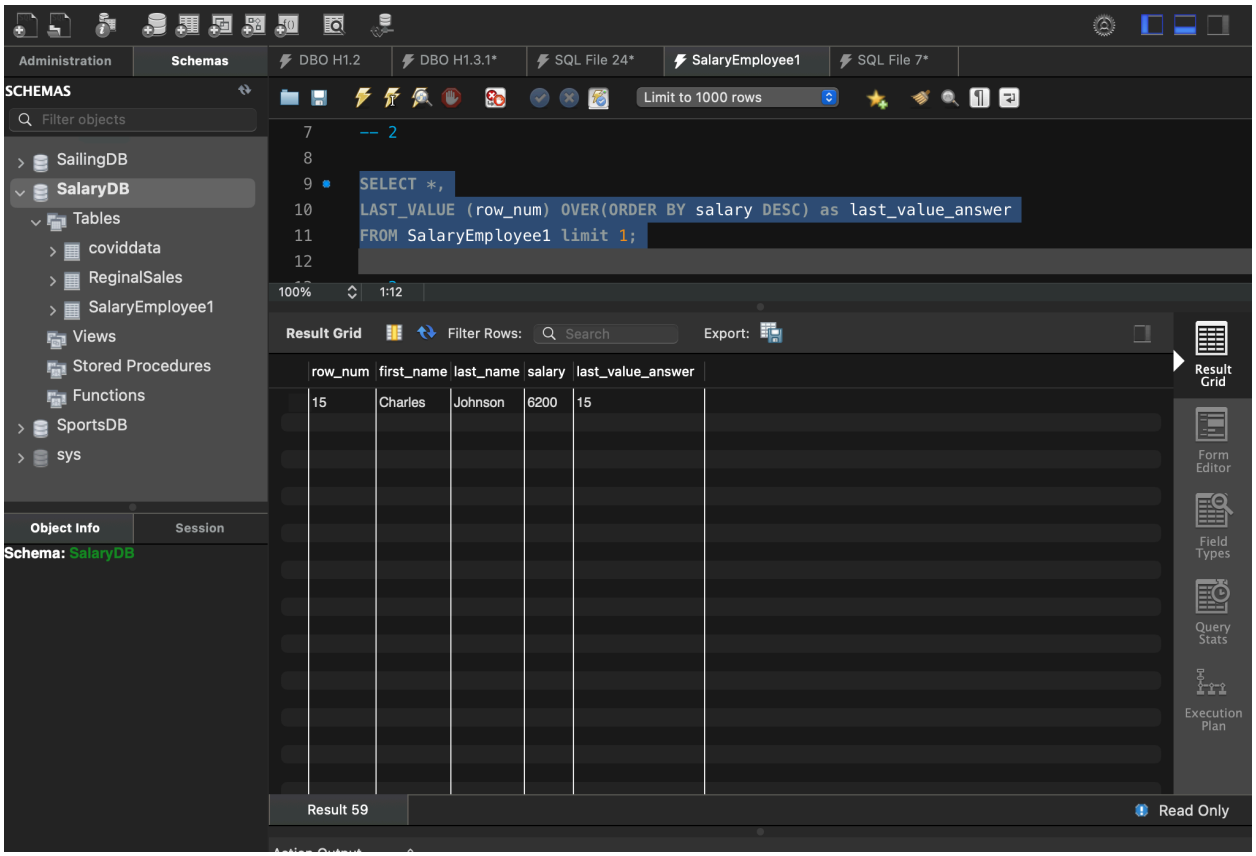
   Result:



2. **SQL Statement**: Write a query to compute for the **LAST_VALUE()** and return the value along with the entire row.

   SQL Query: SELECT *,
   LAST_VALUE (row_num) OVER(ORDER BY salary DESC) as last_value_answer
   FROM SalaryEmployee1 limit 1;

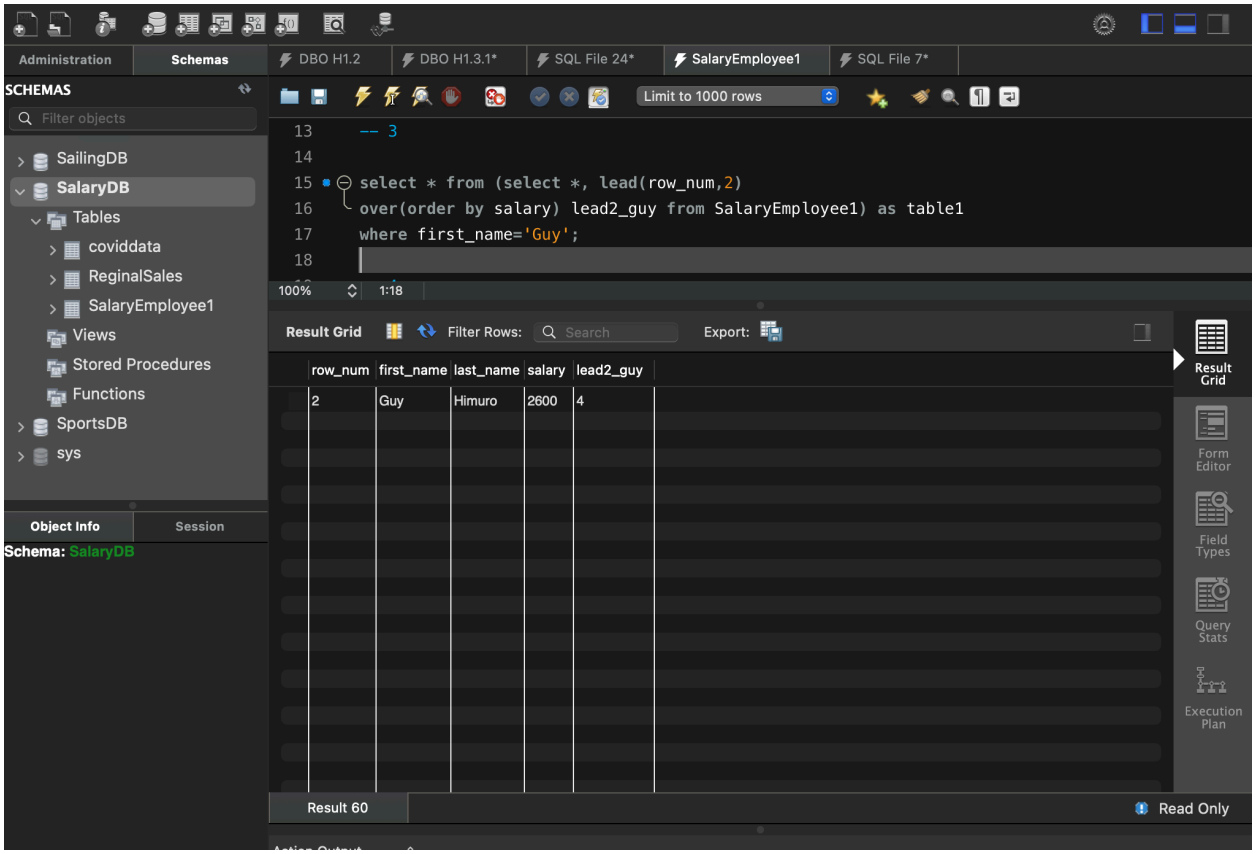
   Result:



3. **SQL Statement:** Write a query to compute for **LEAD(2)** for Guy and return the value along with the Guy's row.

   SQL Query: select * from (select *, lead(row_num,2)
   over(order by salary) lead2_guy from SalaryEmployee1) as table1
   where first_name='Guy';

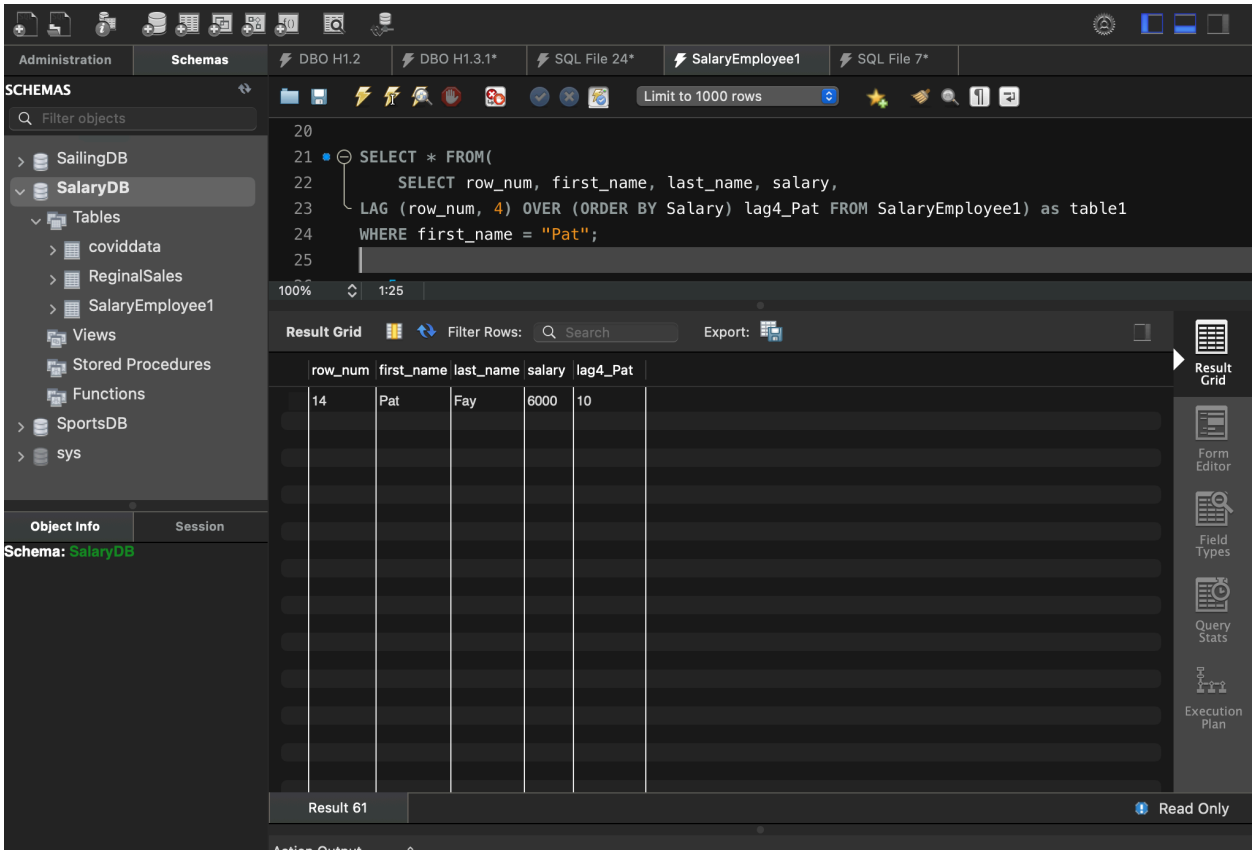
   Result:



4. **SQL Statement:** Write a query to compute for **LAG(4)** for Pat and return value along with Pat's row.

   SQL Query: SELECT * FROM(
   SELECT row_num, first_name, last_name, salary,
   LAG (row_num, 4) OVER (ORDER BY Salary) lag4_Pat FROM SalaryEmployee1) as table1
   WHERE first_name = "Pat";

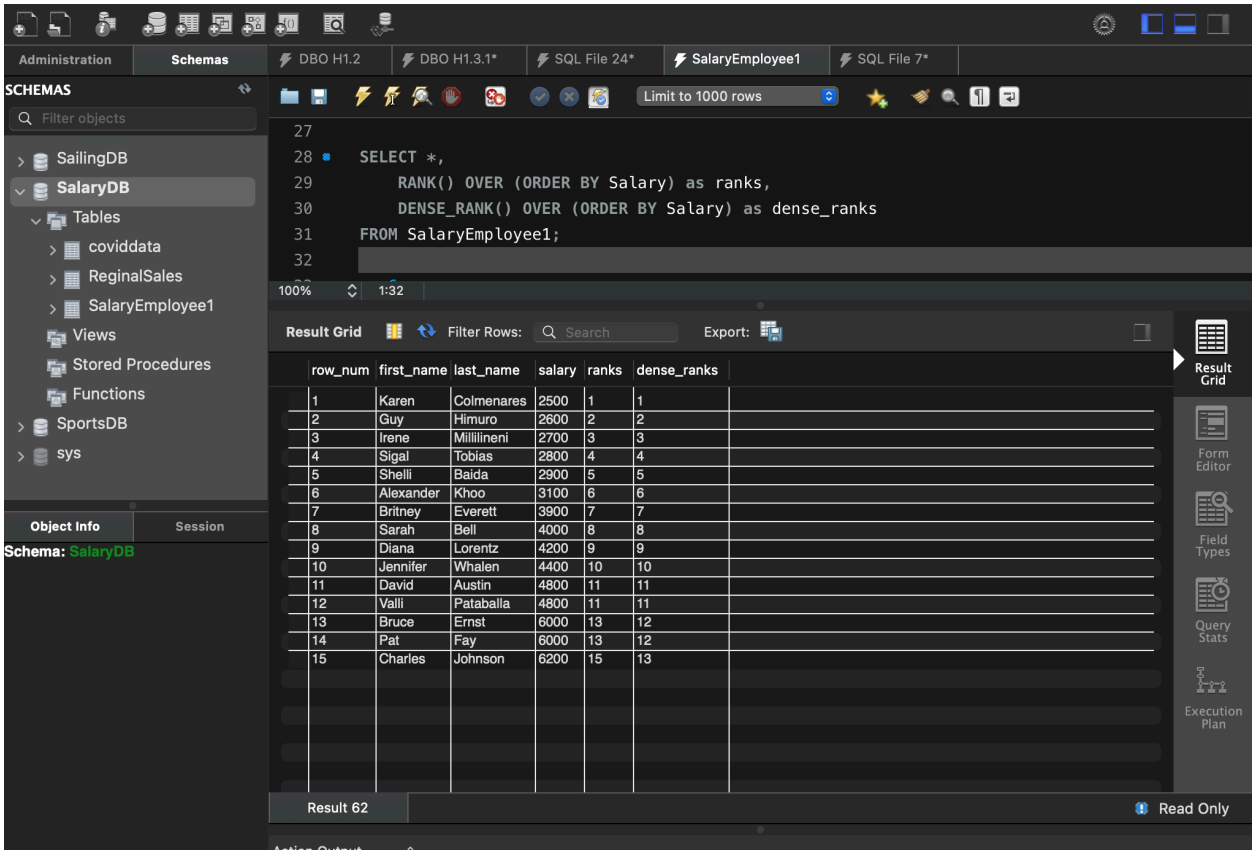
   Result:



5. **SQL Statement**: Write a query to compute the **RANK()** and **DENSE_RANK()** and return the entire dataset, including the rank and dense rank for each employee.

   SQL Query: SELECT *,
   RANK() OVER (ORDER BY Salary) as ranks,
   DENSE_RANK() OVER (ORDER BY Salary) as dense_ranks
   FROM SalaryEmployee1;

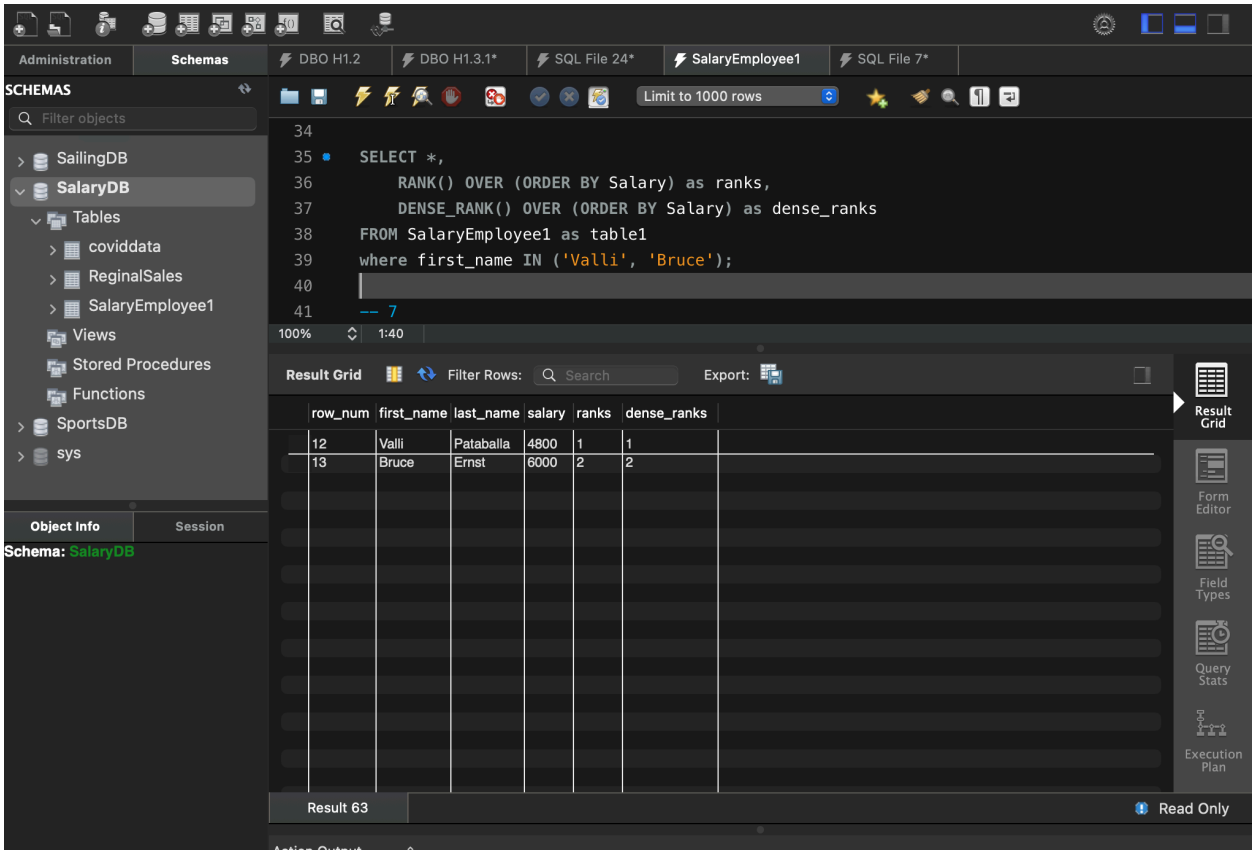
   Result:



6. **SQL Statement**: Write a query to compute the **RANK()** and **DENSE_RANK()** but only return Valli's and Bruce's rank and dense rank.

SQL Query: SELECT *,
RANK() OVER (ORDER BY Salary) as ranks,
DENSE_RANK() OVER (ORDER BY Salary) as dense_ranks
FROM SalaryEmployee1 as table1
where first_name IN ('Valli', 'Bruce');

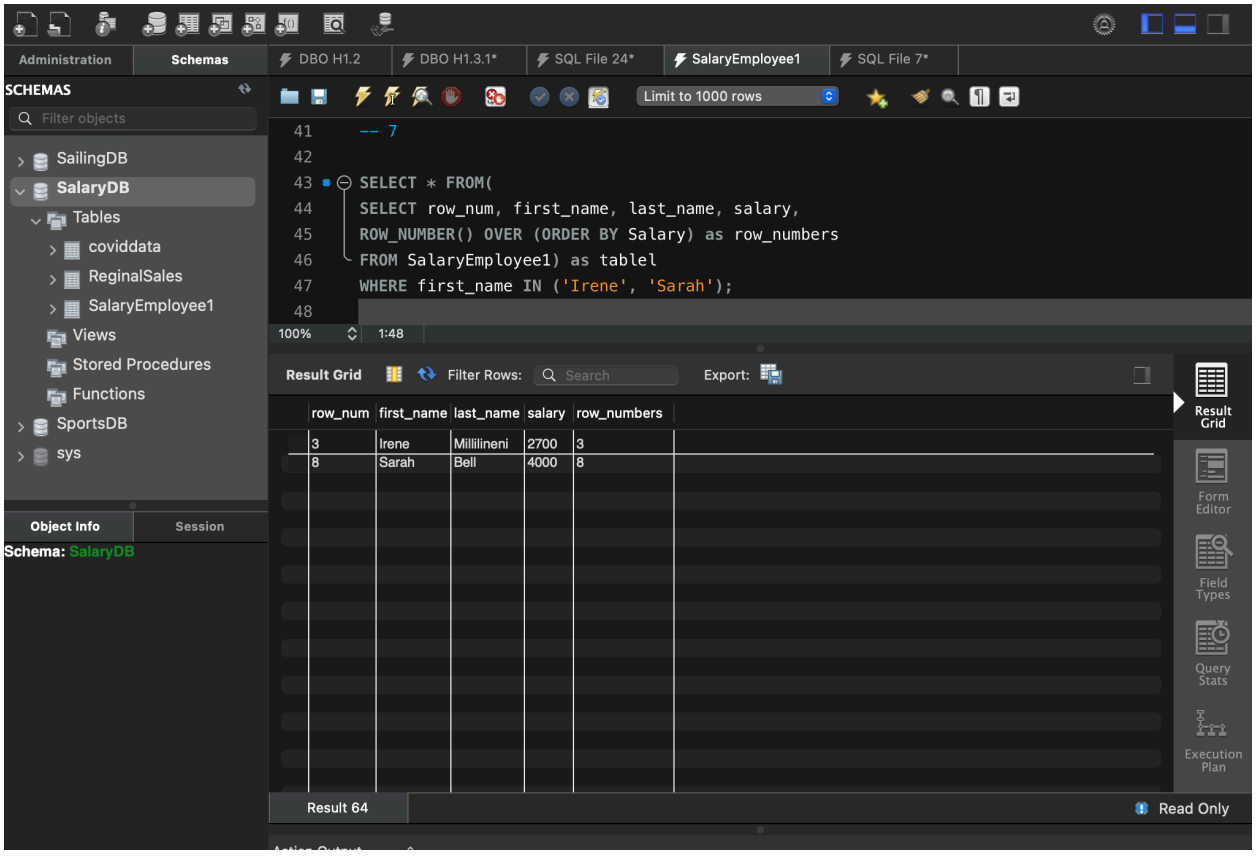
Result:



7. **SQL Statement**: Write a query to compute the **ROW_NUMBER()** for Irene and Sarah and only return the rows corresponding to them.

   SQL Query: SELECT * FROM(
   SELECT row_num, first_name, last_name, salary,
   ROW_NUMBER() OVER (ORDER BY Salary) as row_numbers
   FROM SalaryEmployee1) as table1
   WHERE first_name IN ('Irene', 'Sarah');

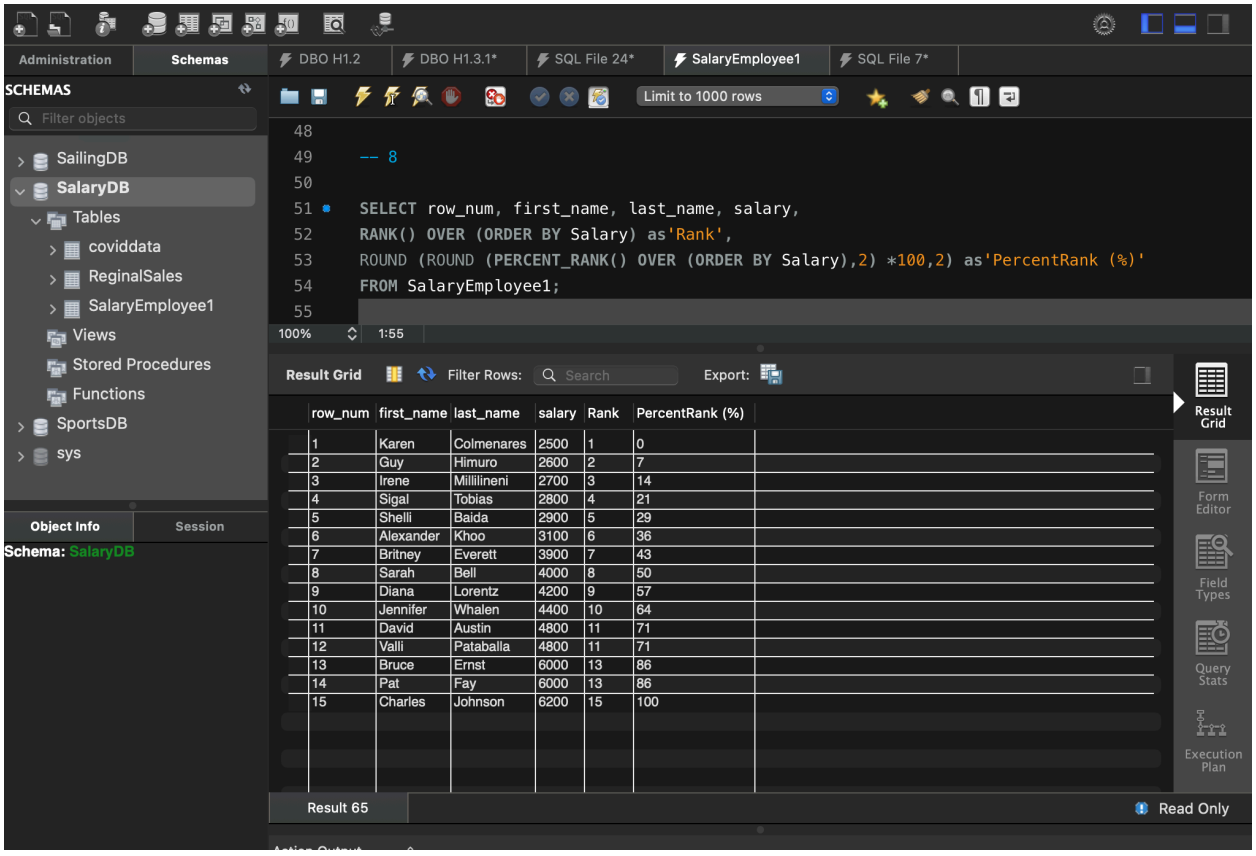
   Result:



8. **SQL Statement**: Write a query to compute the **PERCENT_RANK()** and return the entire dataset, including the percent rank for each employee. Format your PERCENT_RANK() values to 100%.

   SQL Query: SELECT row_num, first_name, last_name, salary,
   RANK() OVER (ORDER BY Salary) as'Rank',
   ROUND (ROUND (PERCENT_RANK() OVER (ORDER BY Salary),2) *100,2)
   as'PercentRank (%)'
   FROM SalaryEmployee1;

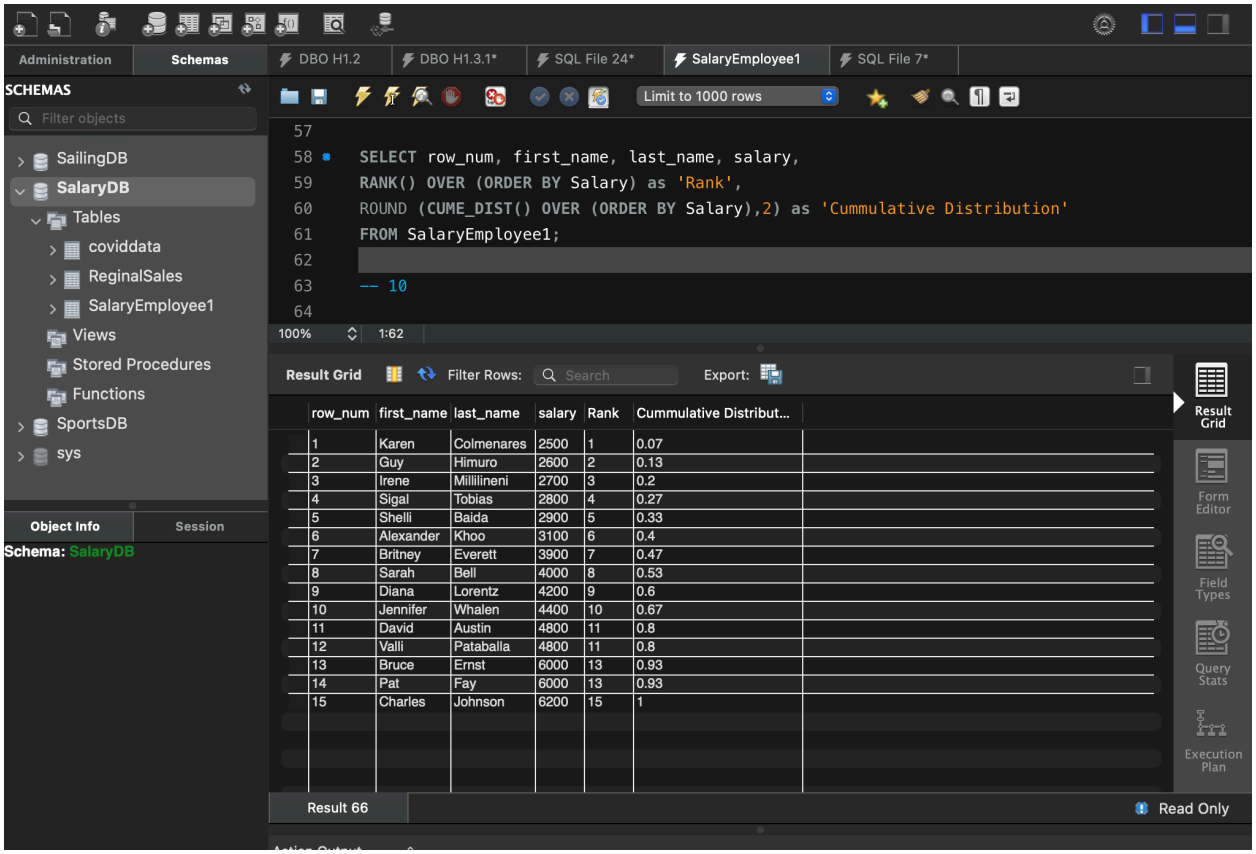
   Result:



9. **SQL Statement**: Write a query to compute the **CUME_DIST()** and return the entire dataset, including the percentage rank for each employee. Format your CUME_DIST() values to 2 decimal places.

   SQL Query: SELECT row_num, first_name, last_name, salary,
   RANK() OVER (ORDER BY Salary) as 'Rank',
   ROUND (CUME_DIST() OVER (ORDER BY Salary),2) as 'Cummulative Distribution'
   FROM SalaryEmployee1;

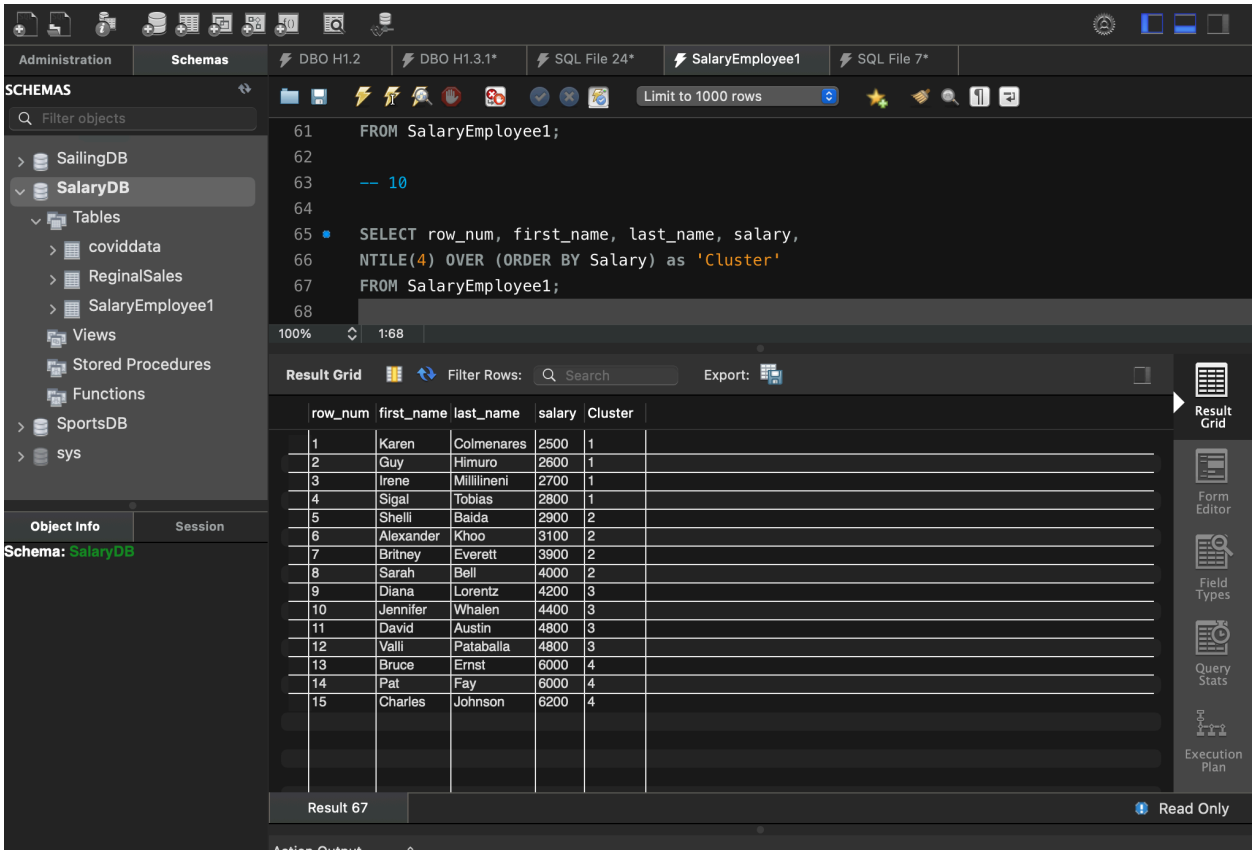
   Result:



10. **SQL Statement**: Write a query to compute the **NTILE(4)** and return the entire dataset showing approximately equal groups/buckets.

    SQL Query: SELECT row_num, first_name, last_name, salary,
    NTILE(4) OVER (ORDER BY Salary) as 'Cluster'
    FROM SalaryEmployee1;

    Result:



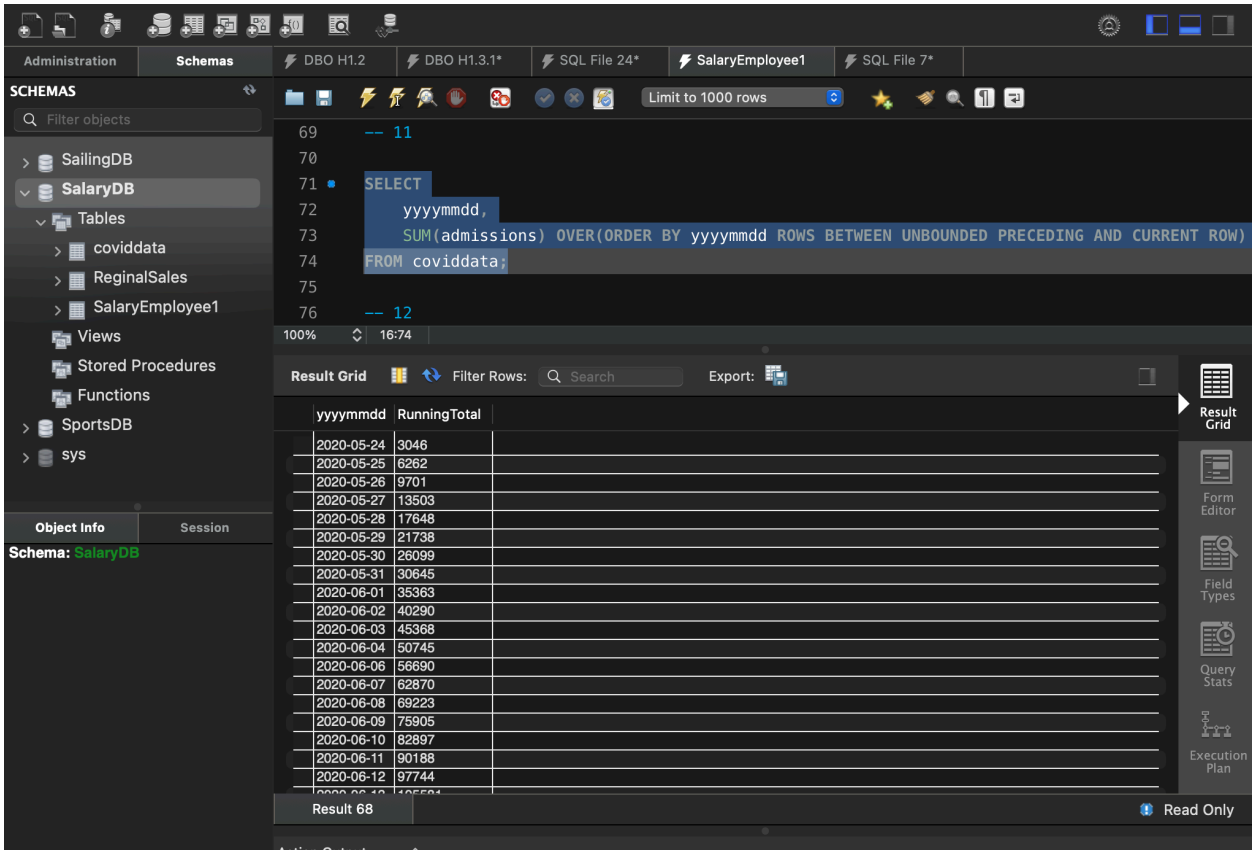
Windowing Activity:

I. **SQL Statement:** Find a running total of covid admissions over three days (i.e., previous day, current day and next day)

SQL Query:

```
SELECT
        yyyymmdd,
        SUM(admissions) OVER(ORDER BY yyyymmdd ROWS BETWEEN
UNBOUNDED PRECEDING AND CURRENT ROW) AS RunningTotal
FROM coviddata;
```

Result:

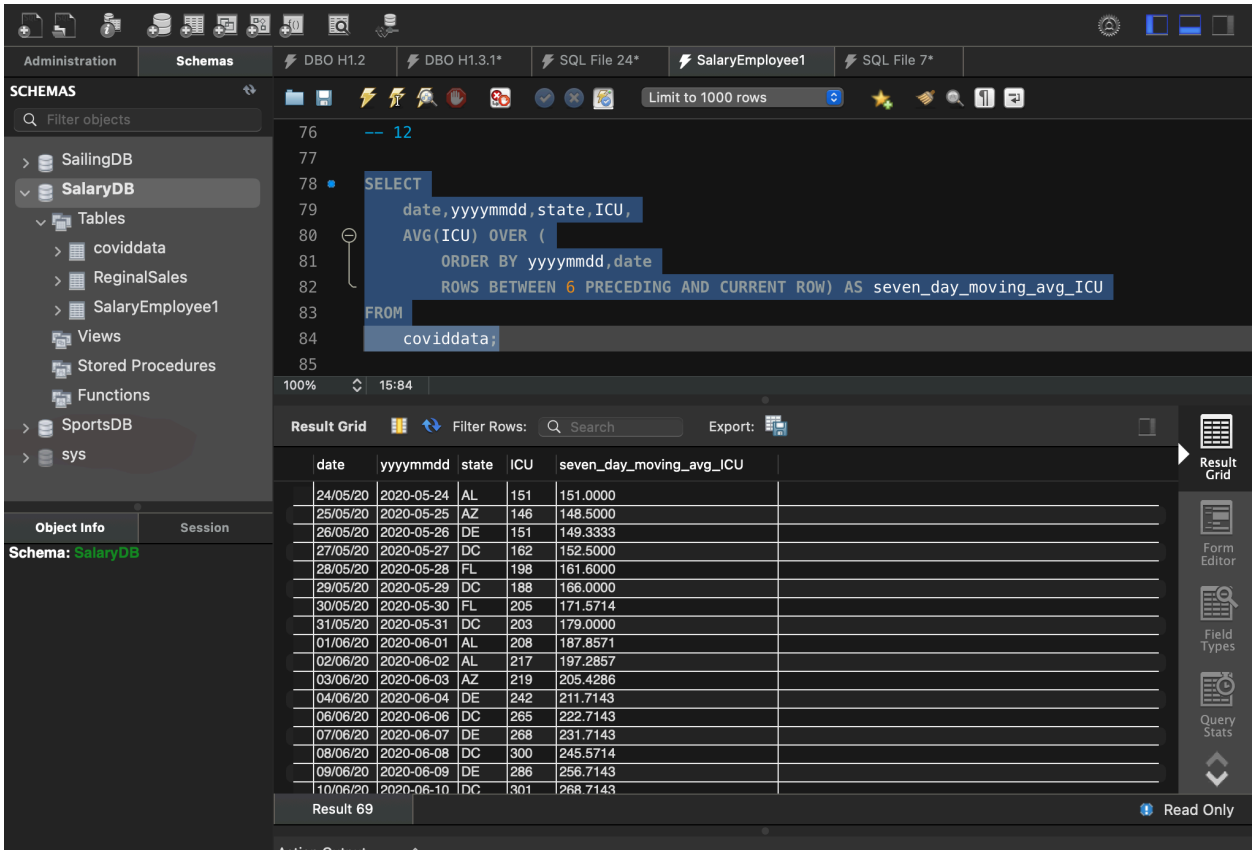


II. **SQL Statement:** Find a 7-day moving average of covid patients in ICU.

SQL Query:

```
SELECT
   date,yyyymmdd,state,ICU,
   AVG(ICU) OVER (
      ORDER BY yyyymmdd,date
      ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS
seven_day_moving_avg_ICU
FROM
   coviddata;
```

Result:



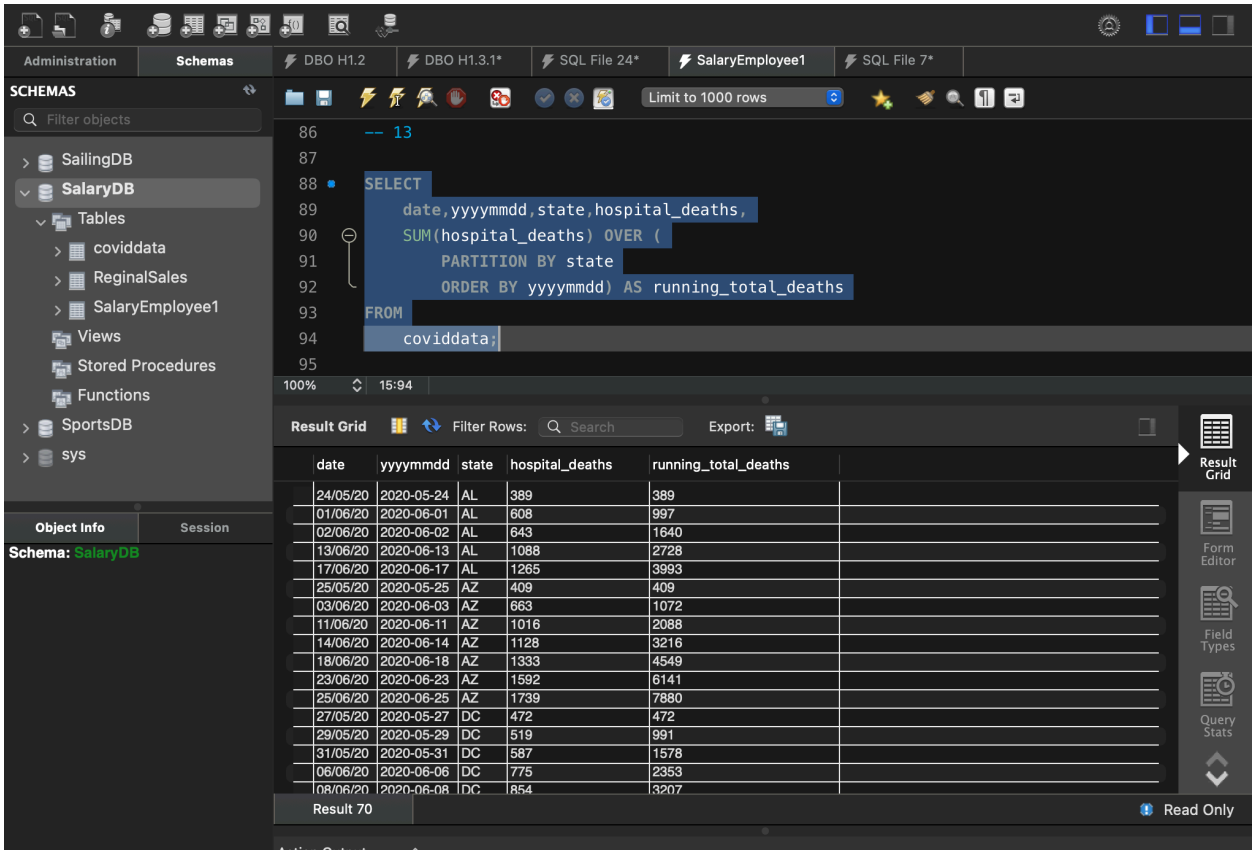
date	yyyymmdd	state	ICU	seven_day_moving_avg_ICU
24/05/20	2020-05-24	AL	151	151.0000
25/05/20	2020-05-25	AZ	146	148.5000
26/05/20	2020-05-26	DE	151	149.3333
27/05/20	2020-05-27	DC	162	152.5000
28/05/20	2020-05-28	FL	198	161.6000
29/05/20	2020-05-29	DC	188	166.0000
30/05/20	2020-05-30	FL	205	171.5714
31/05/20	2020-05-31	DC	203	179.0000
01/06/20	2020-06-01	AL	208	187.8571
02/06/20	2020-06-02	AL	217	197.2857
03/06/20	2020-06-03	AZ	219	205.4286
04/06/20	2020-06-04	DE	242	211.7143
06/06/20	2020-06-06	DC	265	222.7143
07/06/20	2020-06-07	DE	268	231.7143
08/06/20	2020-06-08	DC	300	245.5714
09/06/20	2020-06-09	DE	286	256.7143
10/06/20	2020-06-10	DC	301	268.7143

III. **SQL Statement:** Find a running total of covid deaths in each state

SQL Query:

```
SELECT
   date,yyyymmdd,state,ICU,
   AVG(ICU) OVER (
      ORDER BY yyyymmdd,date
      ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS
seven_day_moving_avg_ICU
FROM
   coviddata;
```

Result:



date	yyyymmdd	state	hospital_deaths	running_total_deaths
24/05/20	2020-05-24	AL	389	389
01/06/20	2020-06-01	AL	608	997
02/06/20	2020-06-02	AL	643	1640
13/06/20	2020-06-13	AL	1088	2728
17/06/20	2020-06-17	AL	1265	3993
25/05/20	2020-05-25	AZ	409	409
03/06/20	2020-06-03	AZ	663	1072
11/06/20	2020-06-11	AZ	1016	2088
14/06/20	2020-06-14	AZ	1128	3216
18/06/20	2020-06-18	AZ	1333	4549
23/06/20	2020-06-23	AZ	1592	6141
25/06/20	2020-06-25	AZ	1739	7880
27/05/20	2020-05-27	DC	472	472
29/05/20	2020-05-29	DC	519	991
31/05/20	2020-05-31	DC	587	1578
06/06/20	2020-06-06	DC	775	2353
08/06/20	2020-06-08	DC	854	3207

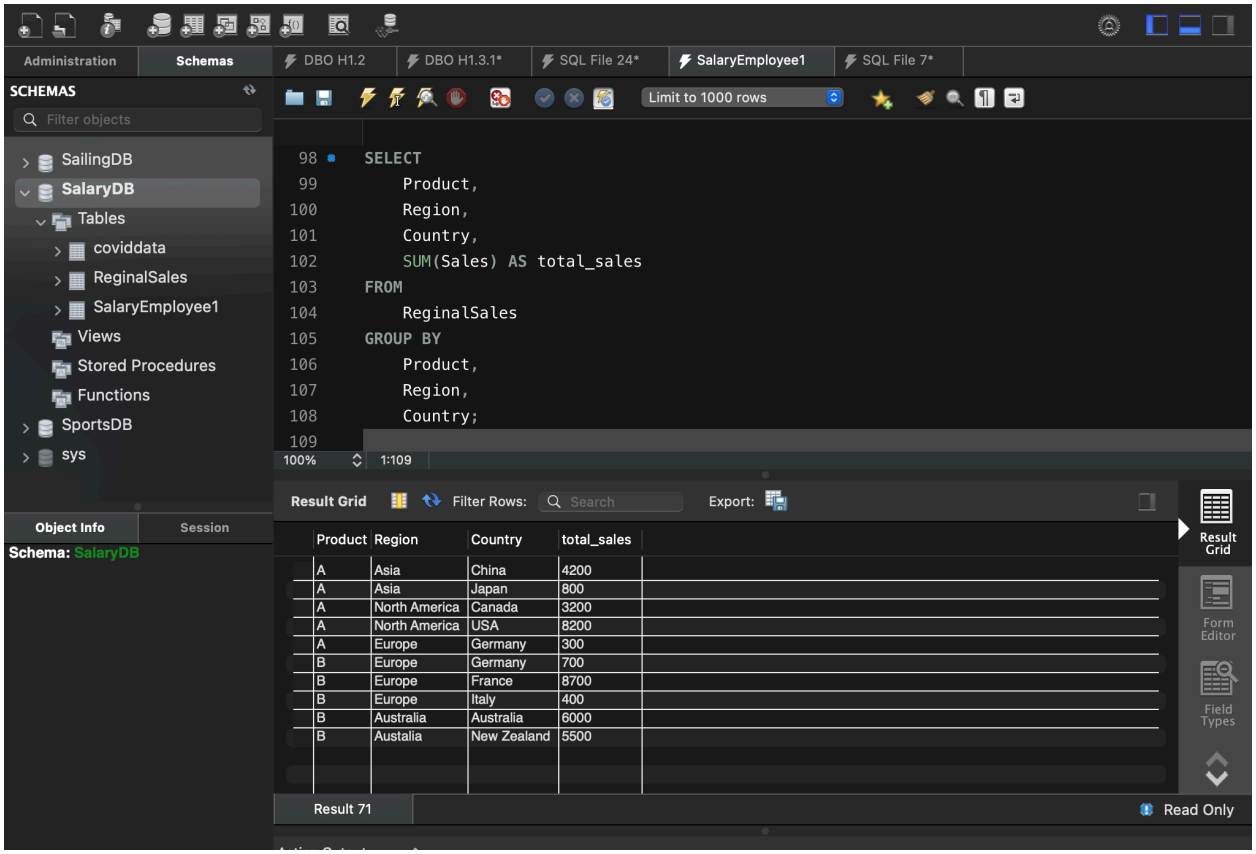
Part 2 : OLAP Queries

I. **SQL Statement:** What are the total sales of each product in each region and country?

SQL Query:

```
SELECT
    Product,
    Region,
    Country,
    SUM(Sales) AS total_sales
FROM
    ReginalSales
GROUP BY
    Product,
    Region,
    Country;
```

Result:

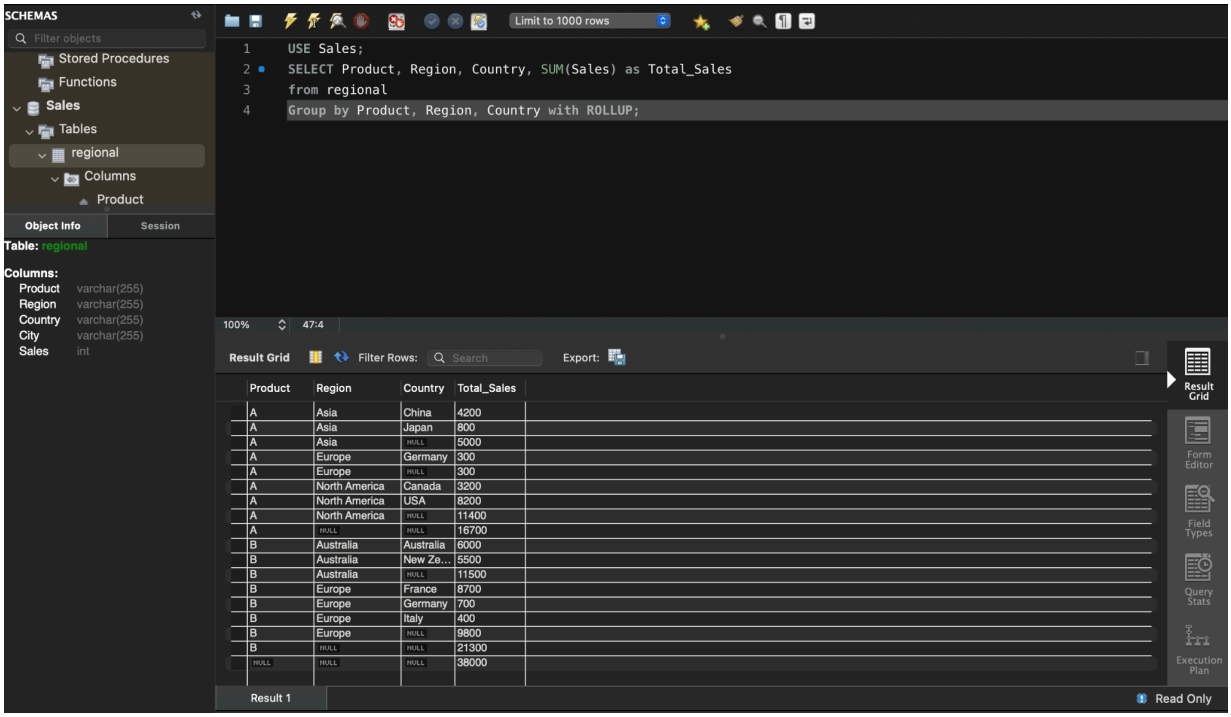


II. **SQL Statement:** What are the total sales of each product in each region and country, including subtotals and the grand total?

SQL Query:

select Product,Region,Country,sum(Sales) totalsales from Regional sales group by Product,Region,country with rollup;

Result:



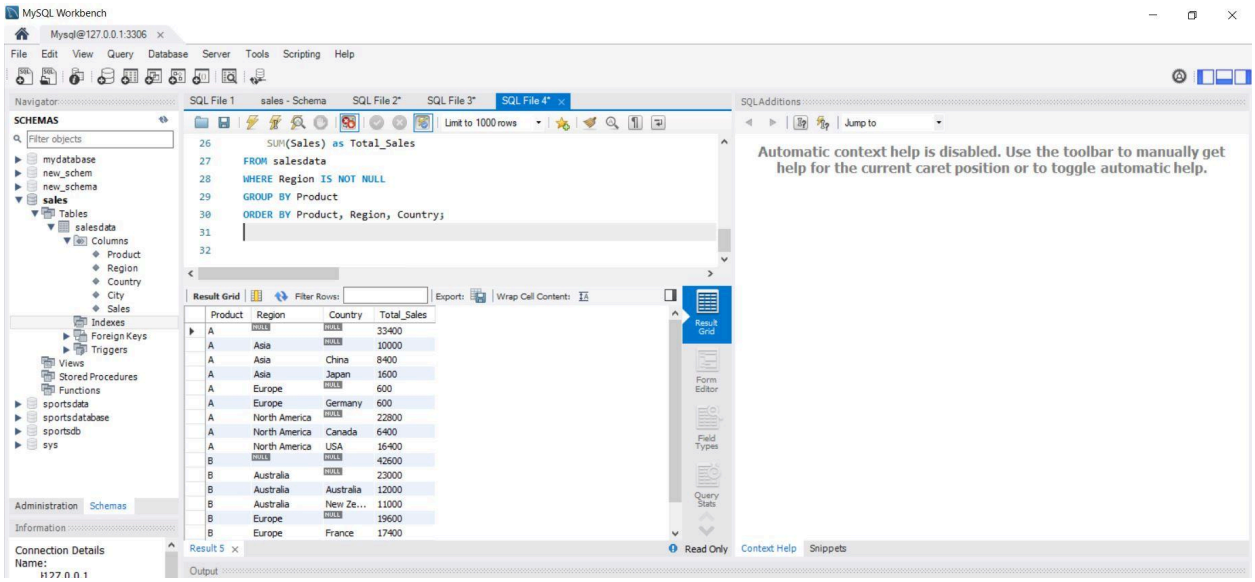
III. **SQL Statement:** What are the total sales generated by each product in each region and country, including all subtotals and grand totals?

SQL Query:

```
SELECT
    Product,
    Region,
    Country,
    SUM(Sales) as Total_Sales
FROM salesdata
GROUP BY Product, Region, Country
UNION ALL
-- Subquery for Product, Region level totals
SELECT
    Product,
    Region,
    NULL AS Country,
    SUM(Sales) as Total_Sales
FROM salesdata
WHERE Region IS NOT NULL
GROUP BY Product, Region
UNION ALL
-- Subquery for Product level totals
SELECT
    Product,
```

```
    NULL AS Region,
    NULL AS Country,
    SUM(Sales) as Total_Sales
FROM salesdata
WHERE Region IS NOT NULL
GROUP BY Product
ORDER BY Product, Region, Country;
```

Result:



Team Contributions:

S.No	Name	Contribution(%)
1.	Hemanth Thathireddy	20%
2.	Satya Dineswara Reddy Setti	20%
3.	Sai Manasa Basani	20%
4.	Alavalapati Veera manohar Reddy	20%
5.	Ying Zhang	20%